Towards Compositional Hierarchical Scheduling Frameworks on Uniform Multiprocessors

João Pedro Craveiro and José Rufino

DI–FCUL–TR–2012–08

DOI:10455/6891

(http://hdl.handle.net/10455/6891)

December 2012

Published at Docs.DI (http://docs.di.fc.ul.pt/), the repository of the Department of Informatics of the University of Lisbon, Faculty of Sciences.

REVISION HISTORY

Date	Revision no.	Summary of changes
28 Dec. 2012	1	— (first version)
31 Jan. 2013	2	More compact IEEE layout; corrected proofs; more detailed explanations with more figures; extended experiments and results; moved some details to appendix.

Towards Compositional Hierarchical Scheduling Frameworks on Uniform Multiprocessors

João Pedro Craveiro and José Rufino
University of Lisbon, Faculty of Sciences, LaSIGE — Lisbon, Portugal
Email: { jcraveiro, ruf }@di.fc.ul.pt

Abstract

In this report, we approach the problem of defining and analysing compositional hierarchical scheduling frameworks (HSF) upon uniform multiprocessor platforms. For this we propose the uniform multiprocessor periodic resource (UMPR) model for a component interface. We extend previous results (for dedicated uniform multiprocessors, and for compositional HSFs on identical multiprocessors), providing a sufficient test for local schedulability of sporadic task sets under global EDF (GEDF) and guidelines for the complex problem of selecting the virtual platform when abstracting a component. Finally, we present simulation results that provide evidence for the need of future developments within the realm of compositional HSFs on uniform multiprocessors.

Keywords

Compositional analysis; Global EDF; ARINC 653; real-time scheduling; schedulability analysis; hierarchical scheduling frameworks; uniform multiprocessors.

I. Introduction

Hierarchical scheduling is a current topic in the mature real-time scheduling theory discipline. In one of the first works on the topic [1], a hierarchical scheduling framework (HSF) is used to allow coexistence of hard real-time tasks and aperiodic soft real-time requests in multimedia applications. Hierarchical scheduling is also used in mixed-criticality systems — systems providing multiple functionalities who differ in how critical they are for the overall welfare of the system, and in the level of assurance of each one's mandatory certification [2]. A common approach to this certification issue of mixed-criticality systems is enforcing isolation through time and space partitioning (TSP) [3], such as in the ARINC 653 specification [4] used in the aerospace industry: tasks are separated into partitions, which are scheduled by a cyclic executive (according to a predefined schedule); when each partition is active, the respective tasks compete for scheduling based on a priority-based scheduling policy; this is a particular case of an HSF [5].

While traditional approaches focused mainly on two-level hierarchies, hierarchical scheduling also sees application in the virtualization field [6], [7], with nested virtualization for advanced security purposes evidencing the need to support more than two hierarchy levels [8]. This highlights the need for an analysis that is independent of the number of hierarchy levels and supports an arbitrary number thereof: *compositional analysis*; compositionality is the property of a complex system which can be analysed by recursively analysing its components and the way they are composed. In this sense, a component comprises a workload, a scheduler, and a resource supply. Figure 1 illustrates a *compositional HSF*; component $\mathcal{C}_0$ is the *root component*, with subcomponents $\mathcal{C}_1$ and $\mathcal{C}_2$. Component $\mathcal{C}_0$ sees the *component interfaces* of $\mathcal{C}_1$ and $\mathcal{C}_2$, which abstract to $\mathcal{C}_0$ the resource demand by each of the subcomponents, hiding how it is composed (tasks and/or subsubcomponents and their characteristics). For each respective $\mathcal{C}_1$ and $\mathcal{C}_2$, the same interface also serves as its *resource interface*,

Work partially supported by FCT and Égide (PESSOA programme), through the transnational cooperation project SAPIENT; by the EC, through project KARYON (IST-FP7-STREP-288195); and by FCT, through the LaSIGE research unit strategic project (UI 408), the CMU|Portugal program, the Individual Doctoral Grant SFRH/BD/60193/2009, and project READAPT (PTDC/EEI-SCR/3200/2012).

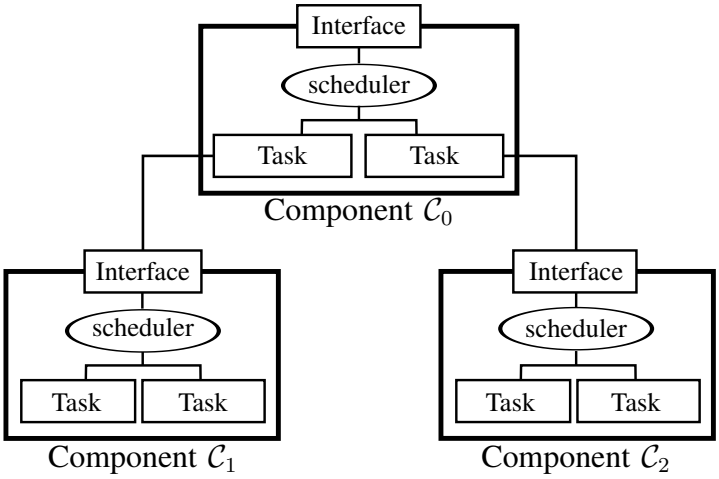


Figure 1. Compositional hierarchical scheduling framework

which expresses the resource supply it is provided, hiding who provides it and how (physical platform vs. parent component).

Compositional analysis comprises three main points [9]:

1) **Local schedulability analysis** Analysing the schedulability of a component's workload upon its scheduler and resource supply.
2) **Component abstraction** Obtaining the component's interface from its inner characteristics.
3) **Interface composition** Transforming a set of component interfaces abstracting the real-time requirements of individual subcomponents into an abstraction of the same type expressing the global requirements of its parent component.

Problem: In some domains of application of HSFs, multiprocessors are already being deployed, but routinely not exploited, because of a lack of support thereto in terms of verification and certification [10], [11]. In [12], we discussed the open problem of extending virtual cluster-based scheduling [13], [14] to uniform multiprocessors (those whose processors differ only in terms of speed), and presented preliminary intuitions. In this report, we advance the state of the art in compositional analysis of HSFs towards the realm of uniform multiprocessors. This is the first paper explicitly dealing with compositional analysis of HSFs on uniform multiprocessor platforms.

Contributions: Besides the uniform multiprocessor resource (UMPR) model to serve as a component interface for compositional analysis (Section III), our contributions w.r.t. the three main points in compositional analysis of HSFs are:

1) **Local schedulability analysis** A sufficient test for sporadic task workloads using global EDF (GEDF) on the UMPR resource model, based on Baruah & Goossens's [15] test for dedicated uniform multiprocessors (Section IV).
2) **Component abstraction** Guidelines for selecting the virtual platform for the UMPR, regarding (for the same total capacity) how close to being identical it should be and its number of processors; we show that simulation confirms these guidelines (Section V).
3) **Interface composition** Simulation results motivating the open questions regarding interface composition and intercomponent scheduling (Section VI).

Due to the added complexity in dealing with uniform multiprocessors, our results on component abstraction and interface composition do not constitute a complete solution thereto.

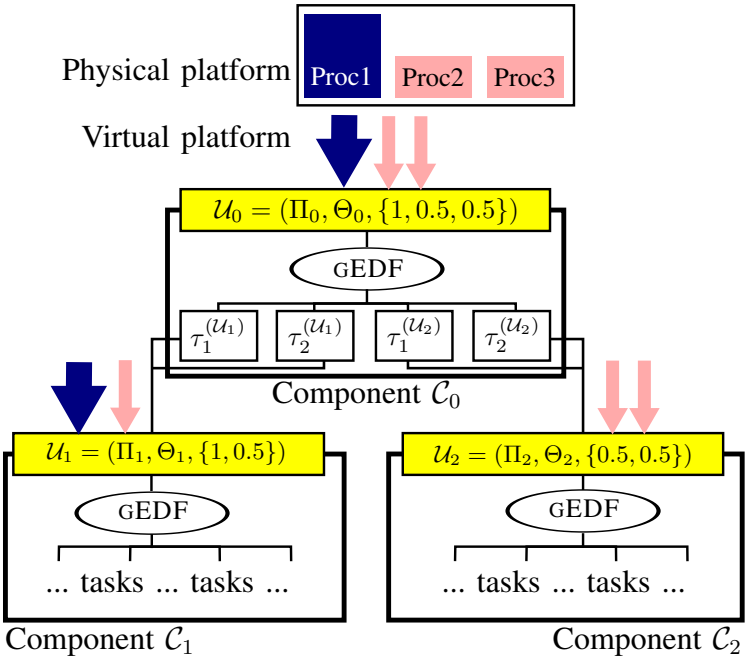


Figure 2. Compositional hierarchical scheduling framework with the UMPR

II. Related work

A. Compositional analysis

Previous works on compositional analysis of multiprocessor HSFs assume a multiprocessor platform with identical unit-capacity processors [13], [14], [16]–[18]. Our work extends to uniform multiprocessors the *multiprocessor periodic resource* model (MPR) [13], [14]; a component interface using the MPR $\Gamma \stackrel{\text{def}}{=} (\Pi, \Theta, m)$ abstracts the provision of Θ processing units over every period with Π time units length over a virtual platform consisting of m identical unit-speed processors.

We chose the MPR because of its simplicity and compositionality potential, and try to maintain as much of these properties as possible by only including in the interface the *schedulable utilization* of each processor in the virtual platform. The remaining works [16]–[18] present a less pessimistic approach (i.e. incur in less overhead), at the expense of abstractions which:

- on the one hand, express the individual *contribution* of each processor;
- on the other hand, hide to the subcomponents' schedulers details of the platform which, in the case of uniform multiprocessors, are needed to guarantee a work-conserving local scheduler (see Section III-A).

B. GEDF *on dedicated uniform multiprocessors*

In [19], Funk et al. provide an exact feasibility test and a sufficient GEDF-schedulability test for implicit-deadline periodic task sets on uniform multiprocessors. Baruah & Goossens [15] provide a sufficient GEDF-schedulability test for constrained-deadline sporadic task sets of uniform multiprocessors. Baruah [20] improves the latter in a way that allows it to be quantitatively evaluated using the processor speedup factor metric.

III. System model

The system we analyze is modeled as a compositional HSF, as pictured in Figure 2. Root component $\mathcal{C}_0$ receives a virtual resource provision directly from the physical platform, whereas the remaining components receive their virtual resource provision from $\mathcal{C}_0$. The tasks in $\mathcal{C}_0$ abstract the resource requirements of the subcomponents for use with classical schedulers and analysis (we

describe these tasks in Section VI). We now define the different parts of our system model, including the resource model we introduce (UMPR).

A. Task and platform models

Task model: A component $\mathcal{C}$ comprises a workload (taskset) τ, with n constrained-deadline sporadic tasks $\tau_i \stackrel{\text{def}}{=} (T_i, C_i, D_i), C_i \leq D_i \leq T_i$, for all $\tau_i \in \tau$. Each task τ_i generates an infinite sequence of instances (jobs) of execution requirement C_i, with two consecutively released jobs separated by *at least* T_i time units; each job must receive C_i units of processor capacity within D_i time units, with no intrajob or intratask parallelism.

We denote by $u_i \stackrel{\text{def}}{=} C_i/T_i$ the *utilization* of task τ_i. The total utilization of taskset τ is defined as $u_{\text{sum}}(\tau) \stackrel{\text{def}}{=} \sum_{i=1}^{n} u_i$. We denote by $\delta_i \stackrel{\text{def}}{=} C_i/D_i$ the *density* of task τ_i. The total density and maximum of density of taskset τ are defined as $\delta_{\text{sum}}(\tau) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \delta_i$ and $\delta_{\max}(\tau) \stackrel{\text{def}}{=} \max_{i=1}^{n} \delta_i$, respectively.

An upper bound to the maximum cumulative execution requirement by jobs of sporadic task τ_i which have both their arrival and deadline times within any time interval with length t is given by the *demand bound function* [21]:

$$\text{dbf}(\tau_i, t) \stackrel{\text{def}}{=} \max\left\{0, \left(\left\lfloor \frac{t - D_i}{T_i} + 1 \right\rfloor\right) \cdot C_i\right\} \ .$$

Platform and scheduling model: We assume a uniform multiprocessor platform with m synchronized processors, defined and represented as

$$\pi \stackrel{\text{def}}{=} \{s_i\}_{i=1}^{m} \ , 1.0 \geq s_i \geq s_{i+1} > 0.0, \text{for all } 1 \leq i < m \ . \tag{1}$$

Each s_i is a non-negative real number representing a processor's normalized relative speed, or *schedulable utilization*; this value corresponds to the amount of processor capacity units it provides within one time unit.

For convenience, we will use $S_\ell(\pi)$, with $\ell \leq m$, to represent the sum of the capacities of the ℓ fastest processors in π: $S_\ell(\pi) \stackrel{\text{def}}{=} \sum_{i=1}^{\ell} s_i$. Hence, $S_m(\pi)$ represents the total capacity of π. We will also make use of the *lambda* parameter, which is defined as

$$\lambda(\pi) \stackrel{\text{def}}{=} \max_{\ell=1}^{m} \frac{S_m(\pi) - S_\ell(\pi)}{s_\ell} \tag{2}$$

and abstracts how different π is from an identical multiprocessor platform. For the extreme case when π *is* an identical multiprocessor platform, $\lambda(\pi) = m - 1$.

We also assume a global EDF scheduling strategy with unrestricted migration, which means a job may be preempted from one processor and resume execution on another processor with negligible migration costs. This scheduling strategy is *work-conserving*, as it operates as follows on a uniform multiprocessor platform: *(i)* if there is an active job waiting to execute, no processor is idle; *(ii)* when the number of active jobs is less than the number of processors, the jobs execute on the fastest processors (and only the slowest ones are idle); *(iii)* greater-priority (e.g., earlier-deadline) jobs execute on the faster processors [15]. Without loss of generality, we refer to global EDF scheduling strategy with unrestricted migration simply as "global EDF" (GEDF).

B. Resource model: the UMPR

To solve the described problem, we propose the uniform multiprocessor periodic resource model.

Definition 1. A *uniform multiprocessor periodic resource* (UMPR) model $\mathcal{U} \stackrel{\text{def}}{=} (\Pi, \Theta, \pi)$ specifies the provision of Θ units of resource over every period of length Π over a virtual uniform multiprocessor platform π (defined as shown in (1)) — with $\Theta \in \mathbb{R}^+$ and $\Pi \in \mathbb{N}$. The ratio Θ/Π is termed *bandwidth*.

In previous works on compositional analysis of HSFs, a *supply bound function* [22] is used in schedulability conditions and to generate component interfaces. The supply bound function of a

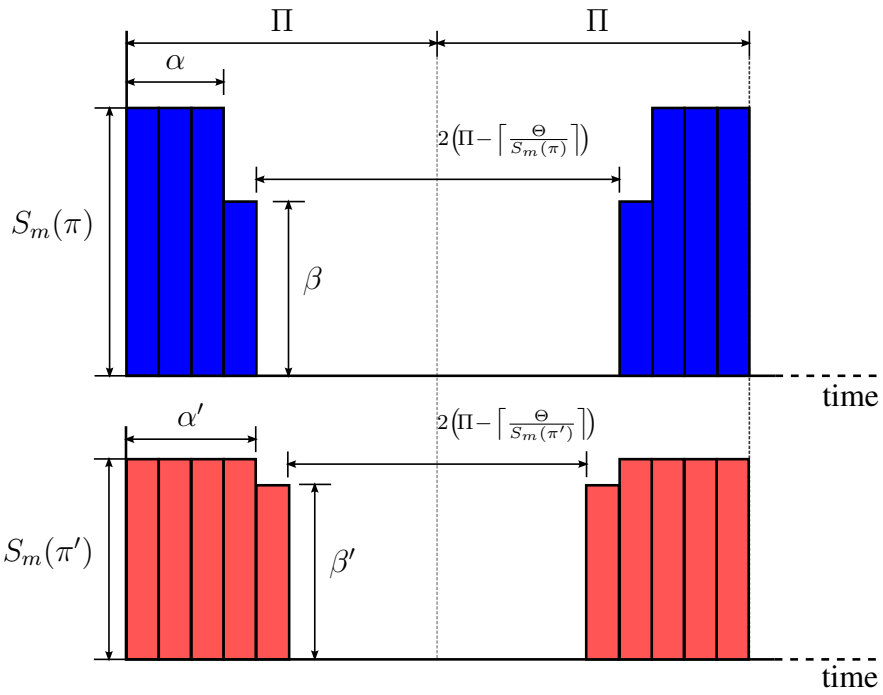


Figure 3. Minimum supply schedule (excerpt) for two UMPRs: $\mathcal{U} = (\Pi, \Theta, \pi)$ (top, blue) and $\mathcal{U}' = (\Pi, \Theta, \pi')$ (bottom, red), where $S_m(\pi) > S_m(\pi')$ (both with m processors)

resource model is a lower bound on the amount of resource supply guaranteed to be given by it in *any* time interval of a given length.

With the MPR, both the number of processors and the total capacity of the virtual platform are expressed in one single abstraction — m — and the role of m in the definition of the supply bound function for the MPR is expressing the total capacity of the platform. In the UMPR, these are separate notions. Figure 3 represents (for two UMPRs with the same bandwidth and number of processors but different total capacity), an excerpt of the schedule which yields the minimum supply in a time interval of length t (as defined in [14]), with the total capacity explicitly denoted as such. The portrayed schedule for $\mathcal{U}$ (resp. $\mathcal{U}'$) can be described as an exclusive supply of π (resp. π') for $\lfloor \frac{\Theta}{S_m(\pi)} \rfloor$ (resp. $\lfloor \frac{\Theta}{S_m(\pi')} \rfloor$) time units, up to one time unit with partial availability (β, resp. β', makes up for the eventual difference to Θ), and no supply for the rest of the period. When this extreme schedule is provided the furthest apart possible on two consecutive periods, we have the schedule which yields the minimum possible supply over any time interval length. Since π' has less capacity than π, $\mathcal{U}'$ corresponds to a greater share of π' than $\mathcal{U}$. Consequently, the maximum interval with no supply is shorter. Hence, with only one minor adaptation, we apply the supply bound function for the MPR [14] to the UMPR; it is defined as

$$\mathrm{sbf}(\mathcal{U}, t) \stackrel{\text{def}}{=} \begin{cases} 0, & t' < 0 \\ w, & t' \geq 0 \wedge x \in [1, y] \\ w - (S_m(\pi) - \beta), & t' \geq 0 \wedge x \notin [1, y] \end{cases} \tag{3}$$

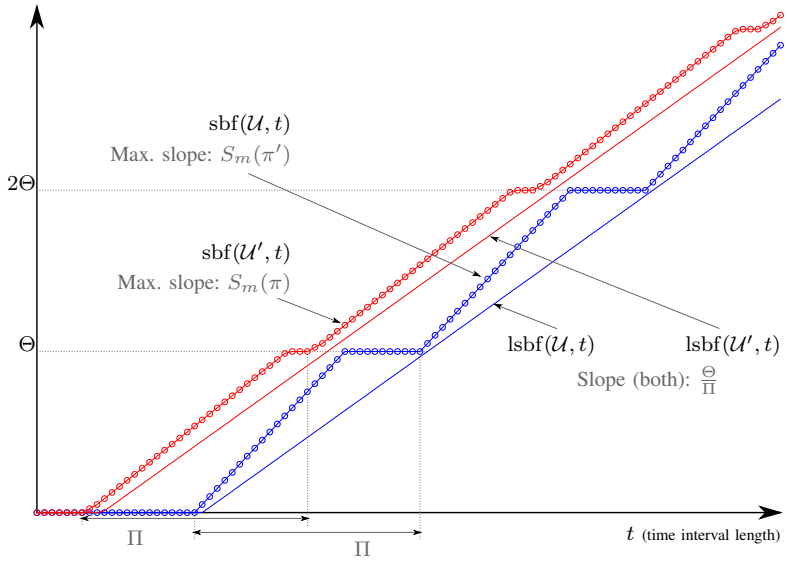


Figure 4. Plot of sbf and lsbf for $\mathcal{U} = (\Pi, \Theta, \pi)$ (blue) and $\mathcal{U}' = (\Pi, \Theta, \pi')$ (red), where $S_m(\pi) > S_m(\pi')$ (both with m processors)

where

$$w = \left\lfloor \frac{t'}{\Pi} \right\rfloor \cdot \Theta + \max\left\{0, S_m(\pi) \cdot x - (S_m(\pi) \cdot \Pi - \Theta)\right\} \ ,$$

$$t' = t - \left(\Pi - \left\lceil \frac{\Theta}{S_m(\pi)} \right\rceil\right) \ ,$$

$$x = \left(t' - \Pi \cdot \left\lfloor \frac{t'}{\Pi} \right\rfloor\right) \ ,$$

$$y = \Pi - \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \ ,$$

$$\text{and } \beta = \Theta - \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot S_m(\pi) \ .$$

When π is an identical multiprocessor platform, $m = S_m(\pi)$, and the supply bound function becomes identical to that of the equivalent MPR.

The same applies to the linear lower bound on the supply bound function, $\mathrm{lsbf}(\mathcal{U}, t)$:

$$\mathrm{lsbf}(\mathcal{U}, t) \stackrel{\text{def}}{=} \frac{\Theta}{\Pi}\left(t - \left(2 \cdot \left(\Pi - \frac{\Theta}{S_m(\pi)}\right) + 2\right)\right) \ . \tag{4}$$

An important property of lsbf is that $\mathrm{lsbf}(\mathcal{U}, t) \leq \mathrm{sbf}(\mathcal{U}, t)$, for all $t > 0$ [14]. Figure 4 shows the plots of sbf and lsbf for the same two UMPRs. Naturally, the range of lengths t for which there is no minimum guaranteed supply is greater for $\mathcal{U}$ than for $\mathcal{U}'$.

IV. Local Schedulability Analysis

To derive a sufficient schedulability condition, we use the framework introduced in [23] and used in many works thereafter [14], [15], [20], [24]. Our approach extends the one followed for dedicated uniform multiprocessors in [15]

Figure 5 shows the considered execution pattern. We consider a job of a task τ_k, and suppose it is the first job to miss a deadline when component $\mathcal{C}$ schedules taskset τ under GEDF using the UMPR model — which corresponds to a time-shared availability of a uniform multiprocessor π.

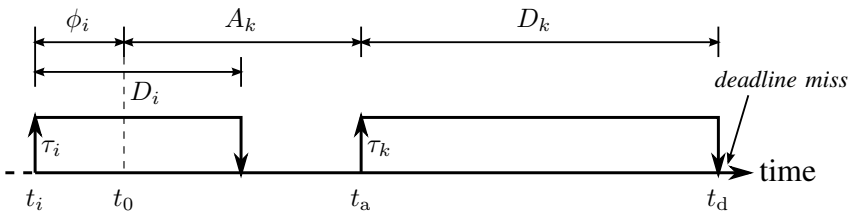


Figure 5. Considered execution pattern, where a job of task τ_k, arriving at instant t_a, is the first to miss a deadline (at instant t_d)

We now follow the strategy used in [15], with the due adaptation to the fact that we are dealing with a potentially non-dedicated resource supply. For this, let us introduce a function $\mathrm{su}(\mathcal{C}, t_1, t_2)$, which denotes the *effective* resource supply that component $\mathcal{C}$ receives over the *specific* time interval $[t_1, t_2[$. This is different from the supply bound function, which denotes the *minimum* guaranteed resource supply the component receives over *any* time interval with a given length, according to its resource interface. Although we do not have a tractable way to compute this quantity, we acknowledge its existence and will take advantage of its following properties:

- $\mathrm{su}(\mathcal{C}, t_1, t_2) + \mathrm{su}(\mathcal{C}, t_2, t_3) = \mathrm{su}(\mathcal{C}, t_1, t_3)$, for all values of t_1, t_2, t_3 such that $t_1 < t_2 < t_3$;
- $\mathrm{sbf}(\mathcal{U}, t_2 - t_1) \leq \mathrm{su}(\mathcal{C}, t_1, t_2)$, for all values of t_1, t_2 such that $t_1 < t_2$; we will call upon this property when deriving our schedulability condition.

A. Interference interval

The time interval we need to consider using this execution pattern is $[t_0, t_d[$. Instant t_d is the absolute deadline that τ_k misses. We now determine which specific instant is t_0. For this, let $W(t)$ (with $t \leq t_d$) denote the cumulative job execution requirement for time interval $[t, t_d[$; this is different from the demand bound function, which is an upper bound on one of the source of this job execution requirement, as we will see in Section IV-B.

Let $I_{\ell,c}$ denote the total duration, over $[t_a, t_d[$, for which exactly ℓ processors, the slowest of which is s_c, are busy scheduling jobs of component $\mathcal{C}$ (with $0 \leq \ell \leq c \leq m$). By definition of GEDF, τ_i's job must be executing on one of the processors whenever there is an idle processor. Unlike when π is a dedicated platform, we cannot guarantee that τ_i is executing on one of the fastest processors in π, but only that it is executing on one of the fastest processors *available*. So, for each duration $I_{\ell,c}$, the job is guaranteed to be executing on a processor of speed, at least, s_c. Therefore,

$$C_k > \sum_{\ell=1}^{m-1} \sum_{c=\ell}^{m} s_c \cdot I_{\ell,c} \ . \tag{5}$$

The total amount of execution completed on $\mathcal{C}$ over the interval $[t_a, t_d[$ is given by the difference between the resource supply it receives — $\mathrm{su}(\mathcal{C}, t_a, t_d)$ — and the total capacity that was idled although available. For each duration $I_{\ell,c}$, the available capacity that is idled is upper bounded by the capacity of the $m - c$ slowest processors; this results by definition: regardless of ℓ, no processor in $\{s_1, \ldots, s_c\}$ can be available but idle. Since this amount is not sufficient for τ_k's job to meet its

deadline at t_{d}, we have

$$\begin{aligned}
W(t_{\mathrm{a}}) &> \mathrm{su}(\mathcal{C}, t_{\mathrm{a}}, t_{\mathrm{d}}) - \sum_{\ell=1}^{m-1} \sum_{c=\ell}^{m} (S_m(\pi) - S_c(\pi)) \cdot I_{\ell,c} \\
&= \mathrm{su}(\mathcal{C}, t_{\mathrm{a}}, t_{\mathrm{d}}) - \sum_{\ell=1}^{m-1} \sum_{c=\ell}^{m} \frac{S_m(\pi) - S_c(\pi)}{s_c} \cdot s_c \cdot I_{\ell,c} \\
&\quad \text{(by (2))} \\
&\geq \mathrm{su}(\mathcal{C}, t_{\mathrm{a}}, t_{\mathrm{d}}) - \lambda(\pi) \cdot \sum_{\ell=1}^{m-1} \sum_{c=\ell}^{m} s_c \cdot I_{\ell,c} \\
&\quad \text{(by (5))} \\
&> \mathrm{su}(\mathcal{C}, t_{\mathrm{a}}, t_{\mathrm{d}}) - \lambda(\pi) \cdot C_k \\
&\quad \text{(by definition of } \delta_k) \\
&= \mathrm{su}(\mathcal{C}, t_{\mathrm{a}}, t_{\mathrm{d}}) - \lambda(\pi) \cdot \delta_k \cdot D_k \\
&\quad \text{(by definition of } \delta_{\max}(\tau)) \\
&\geq \mathrm{su}(\mathcal{C}, t_{\mathrm{a}}, t_{\mathrm{d}}) - \lambda(\pi) \cdot \delta_{\max}(\tau) \cdot D_k \ . \qquad (6)
\end{aligned}$$

Now let us consider a particular instant t_0, which is formally defined as the smallest value $t \leq t_{\mathrm{a}}$ such that $W(t) \geq \mathrm{su}(\mathcal{C}, t, t_{\mathrm{d}}) - \lambda(\pi) \cdot \delta_{\max}(\tau) \cdot (t_{\mathrm{d}} - t)$. This means that t_0 is the earliest instant t so that, in the time interval $[t, t_{\mathrm{a}}[$, no resource capacity available to $\mathcal{C}$ was idled. As a consequence, we can say that, for an arbitrarily small positive number ϵ, some processor was available to $\mathcal{C}$ but idle in the time interval $[t_0 - \epsilon, t_0[$.

B. Component demand

We have defined the interval we need to consider — $[t_0, t_{\mathrm{d}}[$. Next, we determine the cumulative execution that GEDF needs (but fails) to execute over that interval, denoted by $W(t_0)$. As we have seen, two sources contribute to $W(t_0)$:

- jobs with arrival and deadline times within $[t_0, t_{\mathrm{d}}[$; this contribution includes the execution requirement of τ_k's deadline-missing job, and is upper-bounded by $\sum_{i=1}^{n} \mathrm{dbf}(\tau_i, A_k + D_k)$;
- jobs that arrive before instant t_0 and carry in some execution to time interval $[t_0, t_{\mathrm{d}}[$; by construction of the execution pattern and by definition of the GEDF algorithm, all the considered carry-in jobs have their deadlines at or before t_{d} and do not miss them.

Let us then consider the carry-in jobs. We start by proving that Lemmas 1 and 2 from [15] apply, with due adaptation, to our case. Let $A_k = t_{\mathrm{a}} - t_0$.

Lemma 1. *Each carry-in job has strictly less than* $(A_k + D_k) \cdot \delta_{\max}(\tau)$ *remaining execution requirement at instant* t_0.

Proof: Let us consider a carry-in job of a task $\tau_i \in \tau$, which arrives at instant $t_i < t_0$ and has not completed its execution by instant t_0 (Figure 5). By definition of instant t_0, it must be true that

$$W(t_i) < \mathrm{su}(\mathcal{C}, t_i, t_d) - \lambda(\pi) \cdot \delta_{\max}(\tau) \cdot (A_k + D_k + \phi_i) \ ,$$

where $\phi_i = t_0 - t_i$. On the other hand,

$$W(t_0) \geq \mathrm{su}(\mathcal{C}, t_0, t_d) - \lambda(\pi) \cdot \delta_{\max}(\tau) \cdot (A_k + D_k) \ .$$

Hence, by definition, the amount of work done by GEDF on this component's schedule over $[t_i, t_0[$ is strictly less than $\mathrm{su}(\mathcal{C}, t_i, t_0) - \lambda(\pi) \cdot \delta_{\max}(\tau) \cdot \phi_i$. Now let C_i' denote the amount of execution received by the carry-in job of τ_i over the time interval $[t_i, t_0[$; thus, $C_i - C_i'$ denotes the carry-in execution time of this job at instant t_0. Let $J_{\ell,c}$ denote the total duration, over $[t_i, t_0[$, for which exactly ℓ processors, the slowest of which is s_c, are busy scheduling jobs of this component (with

$0 \leq \ell \leq c \leq m$). By definition of GEDF, τ_i's job must be executing on one of the processors whenever there is an idle processor. Therefore,

$$C'_i \geq \sum_{\ell=1}^{m-1} \sum_{c=\ell}^{m} s_c \cdot J_{\ell,c} \ . \tag{7}$$

The total amount of execution completed on $\mathcal{C}$ over the interval $[t_i, t_0[$ is given by the difference between the resource supply it receives — $\mathrm{su}(\mathcal{C}, t_i, t_0)$ — and the total capacity that was idled although available. For each duration $J_{\ell,c}$, the available capacity that is idled is upper bounded by the capacity of the $m-c$ slowest processors. Since this amount is not sufficient for τ_k's job to meet its deadline at t_d:

$$W(t_i) - W(t_0) > \mathrm{su}(\mathcal{C}, t_i, t_0) - \sum_{\ell=1}^{m-1} \sum_{c=\ell}^{m} (S_m(\pi) - S_c(\pi)) \cdot J_{\ell,c}$$

So we have

$$\begin{aligned}
\mathrm{su}(\mathcal{C}, t_i, t_0) - \lambda(\pi) \cdot \delta_{\max}(\tau) \cdot \phi_i &> \mathrm{su}(\mathcal{C}, t_i, t_0) - \sum_{\ell=1}^{m-1} \sum_{c=\ell}^{m} (S_m(\pi) - S_c(\pi)) \cdot J_{\ell,c} \\
-\lambda(\pi) \cdot \delta_{\max}(\tau) \cdot \phi_i &> - \sum_{\ell=1}^{m-1} \sum_{c=\ell}^{m} \frac{S_m(\pi) - S_\ell(\pi)}{s_c} \cdot s_c \cdot J_{\ell,c} \\
&\text{(by (2))} \\
-\lambda(\pi) \cdot \delta_{\max}(\tau) \cdot \phi_i &> - \lambda(\pi) \cdot \sum_{\ell=1}^{m-1} \sum_{c=\ell}^{m} s_c \cdot J_{\ell,c} \\
&\text{(by (7))} \\
-\lambda(\pi) \cdot \delta_{\max}(\tau) \cdot \phi_i &> - \lambda(\pi) \cdot C'i \\
&\text{(rearranging)} \\
C'_i &> \delta_{\max}(\tau) \cdot \phi_i
\end{aligned}$$

Since $C_i = \delta_i \cdot D_i \leq \delta_{\max}(\tau) \cdot D_i$, we have that

$$\begin{aligned}
C_i - C'_i &< \delta_{\max}(\tau) \cdot D_i - \delta_{\max}(\tau) \cdot \phi_i \\
&= \delta_{\max}(\tau) \cdot (D_i - \phi_i) \ .
\end{aligned} \tag{8}$$

By construction of the considered execution pattern, we have that the absolute deadline of this τ_i's job ($t_i + D_i$) is not greater than t_d, so

$$\begin{aligned}
t_i + D_i - t_0 &\leq t_d - t_0 \\
D_i - \phi_i &\leq A_k + D_k
\end{aligned}$$

Replacing this in (8) we prove the lemma. ■

Lemma 2. *There are, at most, $\nu = m - 1$ carry-in jobs.*

Proof: We use the same principle as Baruah & Goossens [15], but are not able to reach an equally tight bound. Let ϵ denote an arbitrarily small positive number; by definition, $W(t_0 - \epsilon) < \mathrm{su}(\mathcal{C}, t_0 - \epsilon, t_\mathrm{d}) - \lambda(\pi) \cdot \delta_{\max}(\tau) \cdot (A_k + D_k + \epsilon)$, whereas $W(t_0) \geq \mathrm{su}(\mathcal{C}, t_0, t_\mathrm{d}) - \lambda(\pi) \cdot \delta_{\max}(\tau) \cdot (A_k + D_k)$; hence:

$$W(t_0 - \epsilon) - W(t_0) < \mathrm{su}(\mathcal{C}, t_0 - \epsilon, t_0) - \lambda(\pi) \cdot \delta_{\max}(\tau) \cdot \epsilon,$$

which shows some processor was idled (although available to $\mathcal{C}$) in $[t_0 - \epsilon, t_0[$. By definition of GEDF, this means all active jobs were executing; hence, the number of carry-in jobs is, at most, one less than the number of available processors. Since we do not possess a way to know exactly how many processors were available to $\mathcal{C}$ in $[t_0 - \epsilon, t_0[$, we are not able to tighten this bound further than the worst case (m processors available, $m - 1$ carry-in jobs). ■

C. Sufficient local schedulability test

With the conditions proven in Lemmas 1 and 2 we are able to formulate a necessary condition for unschedulability of $\mathcal{C}$, from which we will then derive the sufficient schedulability condition.

Lemma 3. *If a component $\mathcal{C}$, comprising a virtual uniform multiprocessor platform π and a sporadic task set τ, is not schedulable under* GEDF *using UMPR model $\mathcal{U} = (\Pi, \Theta, \pi)$, then for some $\tau_k \in \tau$ and some $A_k \geq 0$*

$$\sum_{i=1}^{n} \mathrm{dbf}(\tau_i, A_k + D_k) + (\nu + \lambda(\pi)) \cdot (A_k + D_k) \cdot \delta_{\max}(\tau) > \mathrm{sbf}(\mathcal{U}, A_k + D_k) \ , \tag{9}$$

Proof: We are now able to establish a strict upper bound on $W(t_0)$, based on the two sources that contribute to such execution requirement. Jobs that have arrival and deadline times inside the interference interval contribute with at most $\sum_{i=1}^{n} \mathrm{dbf}(\tau_i, A_k + D_k)$, while jobs that arrive before instant t_0 and carry in some execution contribute with strictly less than $m \cdot (A_k + D_k) \cdot \delta_{\max}(\tau)$. So we have that

$$W(t_0) < \sum_{i=1}^{n} \mathrm{dbf}(\tau_i, A_k + D_k) + \nu \cdot (A_k + D_k) \cdot \delta_{\max}(\tau) \ .$$

From the definition of t_0, we have that $W(t_0) \geq \mathrm{su}(\mathcal{C}, t_0, t_\mathrm{d}) - \lambda(\pi) \cdot (A_k + D_k) \cdot \delta_{\max}(\tau)$. Applying that and rearranging we obtain the following inequation:

$$\sum_{i=1}^{n} \mathrm{dbf}(\tau_i, A_k + D_k) + (\nu + \lambda(\pi)) \cdot (A_k + D_k) \cdot \delta_{\max}(\tau) > \mathrm{su}(\mathcal{C}, t_0, t_\mathrm{d}) \ .$$

Since we have no tractable way to compute $\mathrm{su}(\mathcal{C}, t_0, t_\mathrm{d})$, we take advantage from the fact that $\mathrm{su}(\mathcal{C}, t_0, t_\mathrm{d}) \geq \mathrm{sbf}(\mathcal{U}, A_k + D_k)$ and prove the lemma. ■

By taking the contrapositive of Lemma 3, we have the following sufficient schedulability condition.

Theorem 1. *A component $\mathcal{C}$ comprising a virtual uniform multiprocessor platform π and n sporadic tasks $\tau \stackrel{\text{def}}{=} \{\tau_i \stackrel{\text{def}}{=} (T_i, C_i, D_i)\}_{i=1}^{n}$ is schedulable under* GEDF *using UMPR model $\mathcal{U} = (\Pi, \Theta, \pi)$, if for all tasks $\tau_k \in \tau$ and all $A_k \geq 0$,*

$$\sum_{i=1}^{n} \mathrm{dbf}(\tau_i, A_k + D_k) + (\nu + \lambda(\pi)) \cdot (A_k + D_k) \cdot \delta_{\max}(\tau) \leq \mathrm{sbf}(\mathcal{U}, A_k + D_k) \ . \tag{10}$$

From Lemma 3 we can also derive a maximum value of A_k we have to consider when verifying the condition.

Corollary 1. *If the necessary condition in (9) holds for some $\tau_k \in \tau$ and some $A_k \geq 0$, then it also holds for a value of A_k such that*

$$A_k < \frac{U + B - D_k \cdot \left(\frac{\Theta}{\Pi} - u_{\mathrm{sum}}(\tau) - 2 \cdot (m-1) \cdot \delta_{\max}(\tau)\right)}{\frac{\Theta}{\Pi} - u_{\mathrm{sum}}(\tau) - 2 \cdot (m-1) \cdot \delta_{\max}(\tau)} \tag{11}$$

where

$$U \stackrel{\text{def}}{=} \sum_{i=1}^{n} (T_i - D_i) \cdot \frac{C_i}{T_i} \ , \qquad \textit{and} \qquad B \stackrel{\text{def}}{=} \frac{\Theta}{\Pi} \cdot \left(2 + 2 \cdot \left(\Pi - \frac{\Theta}{S_m(\pi)}\right)\right) \ .$$

Proof: From *(i)* the necessary condition for unschedulability in (9), *(ii)* the definitions of $\mathrm{dbf}(\tau_i, t)$, ν and $\lambda(\pi)$, and *(iii)* the fact that $\mathrm{sbf}(\mathcal{U}, t) \geq \mathrm{lsbf}(\mathcal{U}, t)$ (for all t), we can derive a necessary condition on A_k as follows.

$$\sum_{i=1}^{n} \mathrm{dbf}(\tau_i, A_k + D_k) + (\nu + \lambda(\pi)) \cdot (A_k + D_k) \cdot \delta_{\max}(\tau) > \mathrm{sbf}(\mathcal{U}, A_k + D_k)$$

$$\Rightarrow \qquad (A_k + D_k) \cdot u_{\mathrm{sum}}(\tau) + U + 2 \cdot (m-1) \cdot (A_k + D_k) > \frac{\Theta}{\Pi}(A_k + D_k) - B \ .$$

Rearranging we obtain the inequality in (11). ■

Due to the pessimism introduced in bounding the number of carry-in jobs (Lemma 2), our test does not generalize the test in [15] for the dedicated uniform multiprocessor case. However, when in the presence of a UMPR which corresponds to a dedicated supply ($\Theta = S_m(\pi) \cdot \Pi$), the latter may be employed instead.

V. Component abstraction (guidelines and experiments)

The technique presented in [14] to generate the MPR for $\mathcal{C}$ consists of: *(i)* assuming Π is specified by the system designer; *(ii)* computing Θ and m so that $\mathcal{C}$ is schedulable with the least possible bandwidth (Θ/Π); component interfaces where bandwidth exceeds the platform capacity are *infeasible*. For the computation of the schedulability test to become tractable, $\mathrm{sbf}(\mathcal{U}, t)$ is replaced by $\mathrm{lsbf}(\mathcal{U}, t)$.

For the UMPR, component abstraction is not this simple, because the notions of number of processors and total capacity are no longer represented together as only m. Even restricting to uniform platforms which are in fact identical, each number of processors $m > 0$ corresponds to an infinite amount of uniform multiprocessor platforms with different total capacities $S_m(\pi) \leq m$, and each total capacity $S_m(\pi)$ may be achieved through any number of identical processors $m \geq S_m(\pi)$. Furthermore, the available physical platform may impose restrictions on this. Because of the evident added complexity, we do not present yet a complete solution for it in this report; we will assume both Π and π are specified, and only Θ is computed to guarantee schedulability. We nevertheless provide important guidelines towards its solution. For this, let us introduce one additional definition.

Definition 2. If a UMPR $\mathcal{U}$ guarantees schedulability of $\mathcal{C}$ (comprising taskset τ) *with its smallest possible bandwidth*, then, for some $\tau_k \in \tau$ and some $A_k \geq 0$,

$$\sum_{i=1}^{n} \mathrm{dbf}(\tau_i, A_k + D_k) + (\nu + \lambda(\pi)) \cdot (A_k + D_k) \cdot \delta_{\max}(\tau) = \mathrm{lsbf}(\mathcal{U}, A_k + D_k) \ .$$

We now provide and prove guidelines regarding the relationship between resource bandwidth and *(i)* how close to being identical the virtual platform is, and; *(ii)* the number of processors. In the second comparison, we vary the number of processors with a fixed total capacity; in the similar considerations done in [14] for the MPR, from which we draw inspiration, this was not possible.

A. Uniform vs. identical multiprocessor platform

We start by showing that platforms with lower values of $\lambda(\pi)$ allow UMPRs with lower resource bandwidth, which leads to a more particular consideration about the relation between (non-identical) uniform multiprocessor platforms and identical ones.

Lemma 4. *Consider UMPR interfaces $\mathcal{U}_1 = (\Pi, \Theta_1, \pi_1)$ and $\mathcal{U}_2 = (\Pi, \Theta_2, \pi_2)$, such that $m_1 = m_2 (= m)$, $S_m(\pi_1) = S_m(\pi_2)$, and $\lambda(\pi_1) < \lambda(\pi_2)$. Suppose each interface guarantees schedulability of the same component $\mathcal{C}$ with its respective smallest possible bandwidth. Then, $\mathcal{U}_2$ has a higher resource bandwidth than $\mathcal{U}_1$, i.e. $\Theta_1 < \Theta_2$.*

Proof: For this lemma and the following ones, we follow the structure of the proof by contradiction of Lemma 2 in [14]. Let us then consider $\mathcal{U}'_2 = (\Pi, \Theta'_2, \pi_2)$ such that $\Theta'_2 \leq \Theta_1$, and suppose $\mathcal{U}'_2$ guarantees schedulability of $\mathcal{C}$ according to Theorem 1.

Let Δ_{d} denote the difference in processor requirements of $\mathcal{C}$ on π_1 and π_2 for some interval of

length $A_k + D_k$ — i.e., the difference between the left sides of (10) for each case:

$$\begin{aligned}\Delta_{\mathrm{d}} = &\left(\sum_{i=1}^{n} \mathrm{dbf}(\tau_i, A_k + D_k) + (\nu_2 + \lambda(\pi_2)) \cdot (A_k + D_k) \cdot \delta_{\max}(\tau)\right) \\ &+ \left(\sum_{i=1}^{n} \mathrm{dbf}(\tau_i, A_k + D_k) - (\nu_1 + \lambda(\pi_1)) \cdot (A_k + D_k) \cdot \delta_{\max}(\tau)\right) \\ &= (\nu_2 + \lambda(\pi_2) - \nu_1 - \lambda(\pi_1)) \cdot (A_k + D_k) \cdot \delta_{\max}(\tau) \ .\end{aligned}$$

By our definition of ν, $\nu_1 = \nu_2$; since $\lambda_1 < \lambda_2$, we have that $\Delta_{\mathrm{d}} > 0$. Let us now obtain the counterpart Δ_{s}, denoting the difference between the linear supply bound functions for $\mathcal{U}_1$ and $\mathcal{U}'_2$, for some interval of length $A_k + D_k$:

$$\begin{aligned}\Delta_{\mathrm{s}} &= \mathrm{lsbf}(\mathcal{U}'_2, A_k + D_k) - \mathrm{lsbf}(\mathcal{U}_1, A_k + D_k) \\ &= \frac{\Theta'_2}{\Pi}\left((A_k + D_k) - \left(2 \cdot \left(\Pi - \frac{\Theta'_2}{S_m(\pi_2)}\right) + 2\right)\right) - \frac{\Theta_1}{\Pi}\left((A_k + D_k) - \left(2 \cdot \left(\Pi - \frac{\Theta_1}{S_m(\pi_1)}\right) + 2\right)\right) \\ &\leq \frac{\Theta_1}{\Pi}\left((A_k + D_k) - \left(2 \cdot \left(\Pi - \frac{\Theta_1}{S_m(\pi_2)}\right) + 2\right)\right) - \frac{\Theta_1}{\Pi}\left((A_k + D_k) - \left(2 \cdot \left(\Pi - \frac{\Theta_1}{S_m(\pi_1)}\right) + 2\right)\right) \\ &\leq \frac{2 \cdot {\Theta_1}^2}{\Pi} \cdot \left(\frac{1}{S_m(\pi_2)} - \frac{1}{S_m(\pi_1)}\right) = 0 \ .\end{aligned}$$

Thus, $\Delta_{\mathrm{s}} \leq 0$. Since $\mathcal{U}_1$ guarantees $\mathcal{C}$ to be schedulable with its smallest possible bandwidth, $\sum_{i=1}^{n} \mathrm{dbf}(\tau_i, A_k + D_k) + (\nu_1 + \lambda(\pi_1)) \cdot (A_k + D_k) \cdot \delta_{\max}(\tau) = \mathrm{lsbf}(\mathcal{U}_1, A_k + D_k)$ for some $A_k + D_k$. Thus, for the same $A_k + D_k$, $\sum_{i=1}^{n} \mathrm{dbf}(\tau_i, A_k + D_k) + (\nu_2 + \lambda(\pi_2)) \cdot (A_k + D_k) \cdot \delta_{\max}(\tau) > \mathrm{lsbf}(\mathcal{U}'_2, A_k + D_k)$. This means that $\mathcal{U}'_2$ *does not* guarantee schedulability of $\mathcal{C}$ according to Theorem 1, which contradicts our initial assumption. ■

Theorem 2. *For the same number of processors and total capacity, UMPRs with (non-identical) uniform multiprocessor virtual platforms are better than those with identical multiprocessor virtual platforms.*

Proof: An identical multiprocessor platform π_1 is a particular case of a uniform multiprocessor platforms where all m_1 processors have the same speed, and $\lambda(\pi_1) = m_1 - 1$. For any other (non-identical) uniform multiprocessor platform π_2, $\lambda(\pi_2) < \lambda(\pi_1)$. Then, the theorem follows from Lemma 4 and its proof. ■

Lemma 4 and Theorem 2 are coherent with previous findings in the literature, which state the superiority of "less identical" uniform multiprocessor platforms to scheduling sporadic task sets [15].

B. Number of processors

We now compare platforms which provide the same total capacity through different number of processors. Since we only want to observe the effect of the number of processors, we need to compare platforms which are equally identical — and for this it does not suffice that they have identical values of $\lambda(\pi)$. We perform two comparations, using one extreme case in each comparison: *(i)* both platforms are identical multiprocessor platforms; *(ii)* both platforms are the furthest possible from being an identical multiprocessor platform ($\lambda(\pi_1) = \lambda(\pi_2) = 0$).

Lemma 5. *Consider UMPR interfaces $\mathcal{U}_1 = (\Pi, \Theta_1, \pi_1)$ and $\mathcal{U}_2 = (\Pi, \Theta_2, \pi_2)$, such that $m_2 = m_1 + 1$, $S_{m_1}(\pi_1) = S_{m_2}(\pi_2)$, $m_1 - 1 = \lambda(\pi_1) < \lambda(\pi_2) = m_2 - 1$. Suppose each interface guarantees schedulability of the same component $\mathcal{C}$ with its respective smallest possible bandwidth. Then, $\mathcal{U}_2$ has a higher resource bandwidth than $\mathcal{U}_1$, i.e. $\Theta_1 < \Theta_2$.*

Proof: The proof is similar to that of Lemma 4, so let us consider again $\mathcal{U}'_2 = (\Pi, \Theta'_2, \pi_2)$ such that $\Theta'_2 \leq \Theta_1$, and suppose $\mathcal{U}'_2$ guarantees schedulability of $\mathcal{C}$ according to Theorem 1. The premises

Table I
EXAMPLE COMPONENTS: $\mathcal{C}_1$ AND $\mathcal{C}_2$

	Taskset	$u_{\mathbf{sum}}$	$\delta_{\mathbf{sum}}$
$\mathcal{C}_1$	$\{(60,5,60), (60,5,60), (60,5,60), (60,5,60), (70,5,70), (70,5,70), (80,5,80), (80,5,80), (80,10,80), (90,5,90), (90,10,90), (90,10,90), (100,10,100), (100,10,100), (100,10,100)\}$	1.304	1.304
$\mathcal{C}_2$	$\{(45,2,40), (45,2,45), (45,3,40), (45,3,45), (50,5,45), (50,5,50), (50,5,50), (50,5,50), (70,5,60), (70,5,60), (70,5,65), (70,5,65), (70,5,65), (70,5,65), (70,5,70)\}$	1.122	1.193

for the computation of necessary conditions on the values of Δ_{d} and Δ_{s} are maintained and we thus have that $\Delta_{\mathrm{d}} > 0$ and $\Delta_{\mathrm{s}} \leq 0$. We are then able to reach the same contradiction and prove Lemma 5. ■

Lemma 6. *Consider UMPR interfaces $\mathcal{U}_1 = (\Pi, \Theta_1, \pi_1)$ and $\mathcal{U}_2 = (\Pi, \Theta_2, \pi_2)$, such that $m_2 > m_1$, $S_{m_1}(\pi_1) = S_{m_2}(\pi_2)$, $\lambda(\pi_1) = \lambda(\pi_2) = 0$. Suppose each interface guarantees schedulability of the same component $\mathcal{C}$ with its respective smallest possible bandwidth. Then, $\mathcal{U}_2$ has a higher resource bandwidth than $\mathcal{U}_1$, i.e. $\Theta_1 < \Theta_2$.*

Proof: The proof is also similar to that of Lemma 4; let us have $\mathcal{U}'_2 = (\Pi, \Theta'_2, \pi_2)$ such that $\Theta'_2 \leq \Theta_1$, and we suppose $\mathcal{U}'_2$ guarantees schedulability of $\mathcal{C}$ according to Theorem 1. The platforms for $\lambda(\pi_1) = \lambda(\pi_2) = 0$ correspond to the extreme cases where [19]:

$$\pi_1 = \{S_{m_1}(\pi_1), \underbrace{0, \ldots, 0}_{m_1 - 1 \text{ processors}}\}, \qquad \pi_2 = \{S_{m_2}(\pi_2), \underbrace{0, \ldots, 0}_{m_2 - 1 \text{ processors}}\}.$$

In this case, $\lambda(\pi_1) = \lambda(\pi_2)$, but $\nu_2 > \nu_1$, so we still have $\Delta_{\mathrm{d}} > 0$. In a similar vein as before, we find that $\Delta_{\mathrm{s}} \leq 0$. We are then able to reach the same type of contradiction regarding the assumptions on $\mathcal{U}'_2$, and prove Lemma 6. ■

The following theorem then follows from Lemmas 5 and 6, and their proofs.

Theorem 3. *UMPRs with virtual platforms providing the same total capacity with a lower number of faster processors are better than those with a greater number of slower processors, provided none of the platforms being compared is "less identical" than the other one.*

This is coherent with Lemma 2 in [14] for the (identical) multiprocessor periodic resource model.

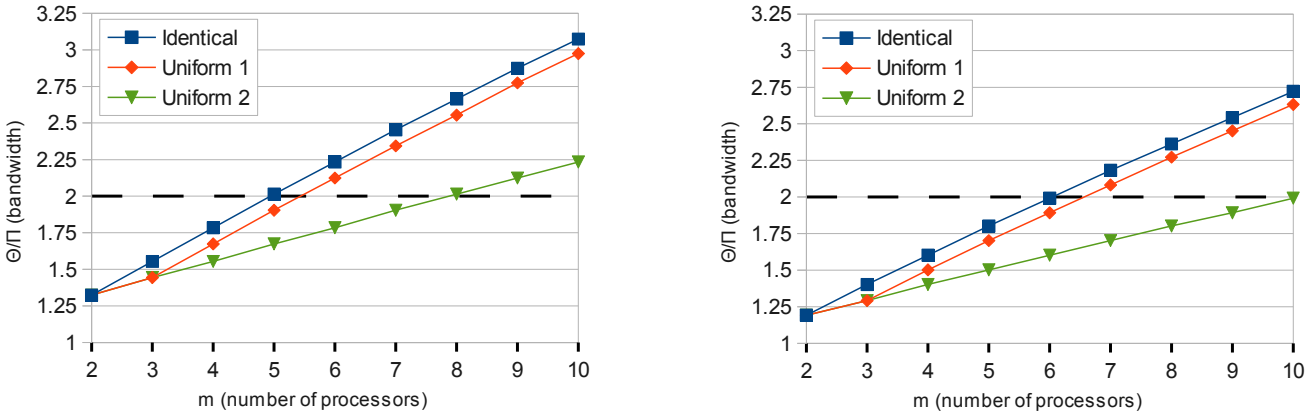
C. Experiments with example components

Let us consider components $\mathcal{C}_1$ and $\mathcal{C}_2$, described in Table I. To allow some comparisons, these samples are taken from [14], and we will select the same values of Π as the authors did there ($\Pi = 6$ for $\mathcal{C}_1$, $\Pi = 5$ for $\mathcal{C}_2$).

1) **Component abstraction guidelines:** To show the effects of the guidelines in Theorems 2 and 3, we generate the minimum UMPR interfaces for three sets of virtual platforms with $2 \leq m \leq 8$ processors, all with total capacity $S_m(\pi) = 2.0$ (unlike with the MPR, we are able to vary the number of processors and the total capacity of the platform independently). In each set, the virtual platform is defined as follows:

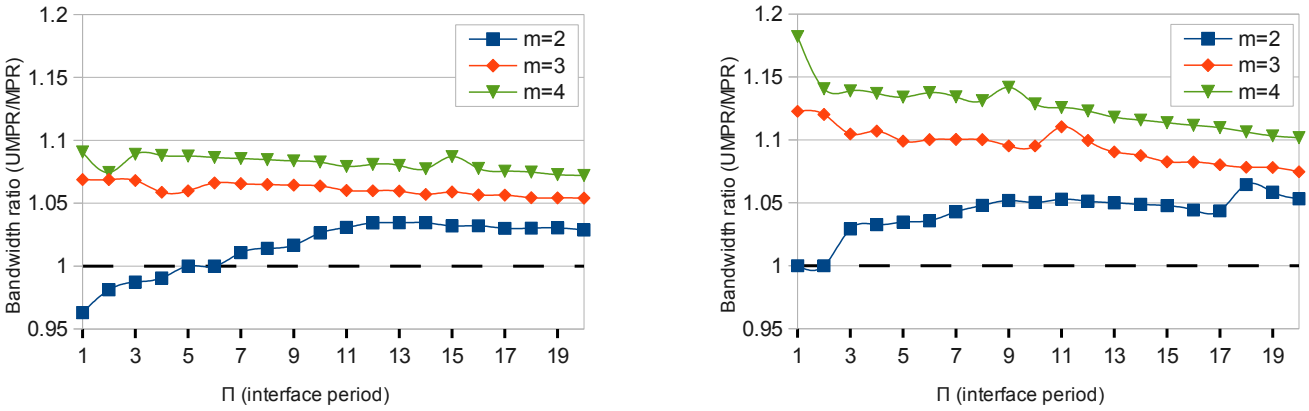
- Identical set: $s_i = 2.0/m$, for all $1 \leq i \leq m$.
- Uniform 1 set: $s_1 = 1.0$, $s_i = 1.0/(m-1)$, for all $2 \leq i \leq m$.
- Uniform 2 set: $s_i = 2.0/(2^{\min\{i, m-1\}})$, for all $1 \leq i \leq m$.

Figure 6 shows the plots of the obtained results, in terms of the bandwidth (Θ/Π) of the computed interfaces. We see that, for within each set of platforms, increasing the number of processors causes an



(a) Component $\mathcal{C}_1$ ($\Pi = 6$) (b) Component $\mathcal{C}_2$ ($\Pi = 5$)

Figure 6. Comparison between minimal bandwidth for UMPRs based on identical and non-identical uniform multiprocessors. In all samples, $S_m(\pi) = 2.0$; points above the dashed line (at $\Theta/\Pi = 2.0$) correspond to infeasible UMPRs)



(a) Component $\mathcal{C}_1$ (b) Component $\mathcal{C}_2$

Figure 7. Ratio between the minimum bandwidth required by a UMPR with an identical multiprocessor platform and by the equivalent MPR (according to each respective schedulability test)

increase in the resource bandwidth of the UMPR interface. This confirms the guideline in Theorem 3, which states that, between two platforms with the same capacity and which are not less identical than one another, virtual platforms with less faster processors are better than those with more slower processors. We also see that, for the same capacity and number of processors, the platforms in the Uniform 1 and Uniform 2 sets yield less resource bandwidth that the one in Identical. This confirms the guideline in Theorem 2, which states that less identical platforms (those with lower values of $\lambda(\pi)$) perform better. It should be noted that, of the resulting UMPRs, those with $\Theta/\Pi > 2.0$ are not feasible.

2) **UMPR vs. MPR:** In another experiment with the same components, we used only identical unit-capacity multiprocessor platforms ($S_m(\pi) = m$). For platforms with 2, 3 and 4 processors, and for each value of Π between 1 and 20, we derived the minimum bandwidth UMPR and MPR interfaces, using, respectively, the schedulability tests in Section IV and in [14]. The plots in Fig. 7 show, for each case, the ratio between the obtained minimum bandwidths (a ratio greater than 1 means the UMPR required more bandwidth than the MPR). We can see that none of the tests is tighter than the other one all of the time, but the MPR test is generally tighter; this arises as a consequence of transitioning from identical to uniform, since we have to incur some pessimism when deriving Eqs. (5) and (7); this is also the case on dedicated uniform multiprocessors [15]. Furthermore, we introduce additional pessimism in the number of carry-in jobs (Lemma 2).

VI. Interface composition and intercomponent scheduling (simulation and open question)

There are no known scheduling algorithms for component interfaces such as the MPR [14] and the UMPR. Then, we need to transform them to tasks than can be scheduled by traditional scheduler (such as GEDF) and analysed with classical results. We now present a method of transforming a UMPR $\mathcal{U}$ into a set of m periodic tasks. A periodic task τ_i is a particular case of a sporadic task whose jobs have their arrival times separated by *exactly* T_i time units.

Definition 3. Consider a UMPR model $\mathcal{U} = (\Pi, \Theta, \pi)$. Let us have $\beta \stackrel{\text{def}}{=} \Theta - S_m(\pi) \cdot \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor$ and $k \stackrel{\text{def}}{=} \max\{\ell : S_\ell(\pi) \leq \beta\}$. We define the transformation from $\mathcal{U}$ to a periodic task set $\tau^{(\mathcal{U})}$ with m tasks $\tau_i^{(\mathcal{U})} \stackrel{\text{def}}{=} (T_i^{(\mathcal{U})}, C_i^{(\mathcal{U})}, D_i^{(\mathcal{U})})$ such that:

- if $i \leq k$, $\tau_i^{(\mathcal{U})} = \left(\Pi, \left(\left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor + 1\right) \cdot s_i, \Pi\right)$;
- if $i = k+1$, $\tau_i^{(\mathcal{U})} = \left(\Pi, \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i + \beta - S_k(\pi), \Pi\right)$;
- otherwise, $\tau_i^{(\mathcal{U})} = \left(\Pi, \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i, \Pi\right)$.

Detailed proofs that *(i)* $\sum_{i=1}^{m} C_i^{(\mathcal{U})} = \Theta$, *(ii)* schedulability of a taskset $\tau^{(\mathcal{U})}$ is a sufficient condition for the supply to $\mathcal{C}$ to be lower bounded by $\mathrm{sbf}(\mathcal{U}, t)$, and *(iii)* if π is in fact an identical multiprocessor platform, this transformation becomes equivalent to that in [14] for the MPR can be found in the Appendix.

Using this transformation, we have performed simulation experiments which reveal the extent of the open problem of intercomponet scheduling in UMPR-based HSFs, whose solution is out of the scope of this report.

A. hsSim and implementation overview

Few real implementations of compositional HSFs exist and they only support uniprocessor [6], [7], [25]. For this reason, we perform simulation using hsSim, an object-oriented hierarchical scheduling simulator, implemented in Java, supporting an arbitrary number of levels [26]. hsSim records a trace of the simulated execution, for visualization with Grasp [27]. To the best of our knowledge, hsSim is the first tool allowing scheduling simulation of compositional HSFs on multiprocessors. In addition to the work described in [26], we have implemented support for uniform multiprocessors in general, and compositional HSFs based on the UMPR in particular. Since the intercomponent scheduling and component–task transformation we consider for now are identical to the ones in [14], hsSim also supports the simulation of compositional HSFs based on the MPR.

For a component abstracted with a UMPR $\mathcal{U}$, we derive the periodic tasks $\tau^{(\mathcal{U})}$ (Definition 3) and add them to the root scheduler. These tasks are implemented as a special type of task (*pseudotask*) which launches jobs of the component (and not the task itself). hsSim features only *one* implementation of the GEDF scheduler, which takes advantage of subtype polymorphism to schedule jobs of all the supported schedulable entities (tasks, components) unaware of their differences. When a job is scheduled on a processor with speed s_i, a binding is created between this processor and the schedulable entity to which the job corresponds[1]. If it is a task, it decreases its remaining execution requirement by s_i. If it is a component, a binding is created between the component's virtual platform and this processor, so that it becomes available to the component's scheduler.

B. Simulation and open question

Now let us consider a compositional HSF with two components, $\mathcal{C}_3$ and $\mathcal{C}_4$, whose internal characteristics we omit (due to paper length constraints and to its irrelevance for the point being made):

[1]Currently, hsSim performs cycle-step execution simulation.

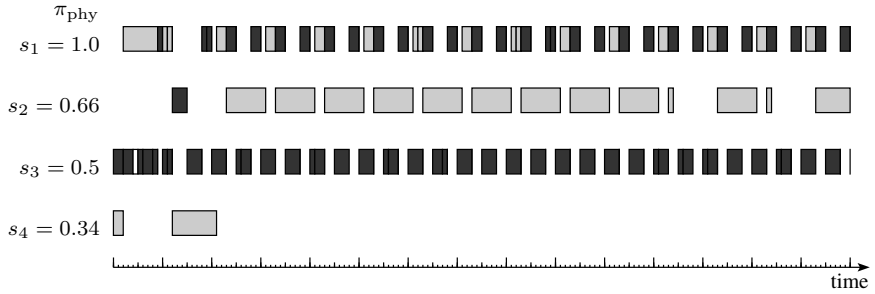


Figure 8. Grasp trace for the compositional HSF consisting of $\mathcal{C}_3$ (tasks traced in dark gray) and $\mathcal{C}_4$ (light gray)

- *Component* $\mathcal{C}_3$ Employs GEDF to schedule a sporadic task set on virtual multiprocessor platform $\pi_3 = \{1.0, 0.66, 0.34\}$; we abstract $\mathcal{C}_3$ using UMPR model $\mathcal{U}_3$, which we transform to a periodic task set $\tau^{(\mathcal{U}_3)}$.
- *Component* $\mathcal{C}_4$ Employs GEDF to schedule a sporadic task set on virtual multiprocessor platform $\pi_4 = \{1.0, 0.5\}$; we abstract $\mathcal{C}_4$ using UMPR model $\mathcal{U}_4$, which we transform to a periodic task set $\tau^{(\mathcal{U}_4)}$.

When we schedule the compositional HSF consisting of $\mathcal{C}_3$ and $\mathcal{C}_4$ over GEDF on a physical platform $\pi_{\text{phy}} = \{1.0, 0.66, 0.5, 0.34\}$, we observe an effect we had anticipated when we formulated the problem in [12] (Fig. 8). We can see that tasks of both components are being scheduled on processors in the physical platform which have no counterpart in the virtual platform for which each of the components was analysed. The observed effect hints that the solution for the question of intercomponent scheduling in compositional HSFs on uniform multiprocessors (based on the UMPR) may require a specific algorithm to schedule UMPRs.

VII. Conclusion

We approached the problem of compositional hierarchical scheduling frameworks (HSF) upon uniform multiprocessor platforms, proposing the Uniform Multiprocessor Resource Model (UMPR) as a component interface. We extended previous related works, providing results and intuitions which span over the three main points of compositional analysis of HSFs: a sufficient local GEDF-schedulability test for sporadic task sets in components with the UMPR as a resource interface; guidelines for the complex problem of selecting the virtual platform when abstracting a component, and; simulation results evidencing the open questions in interface composition/intercomponent scheduling.

Current work includes improving our schedulability test and solving, for the UMPR, the intercomponent scheduling and interface composition questions of compositional analysis. We also envision assessing the potential for other approaches for identical multiprocessors (e.g., [16], [17]) to be applied to uniform multiprocessor platforms (albeit with a different compromise between simplicity/compositionality and resource bandwidth optimality). Finally, we are also working on backporting some of the code of hsSim to add support to compositional analysis support of HSFs to the Cheddar tool [10], [26].

Acknowledgment

The authors thank Insik Shin and Arvind Easwaran (for promptly answering our doubts about specific aspects of the MPR), Pedro M. Ferreira (for comments which improved the readability of this report), and the organizers and attendees of RTSOPS 2012 (for the fruitful discussions there).

REFERENCES

[1] L. Abeni and G. Buttazzo, "Integrating multimedia applications in hard real-time systems," in *RTSS '98*, Madrid, Spain, Dec. 1998.

[2] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, N. Megow, and L. Stougie, "Scheduling real-time mixed-criticality jobs," *IEEE Trans. Comput.*, vol. 61, no. 8, pp. 1140–1152, Aug. 2012.

[3] J. Windsor and K. Hjortnaes, "Time and space partitioning in spacecraft avionics," in *SMC-IT 2009*, Jul. 2009, pp. 13–20.

[4] AEEC, "Avionics application software standard interface, part 1 - required services," Aeronautical Radio, Inc., ARINC Spec. 653P1-2, Mar. 2006.

[5] J. Rufino, J. Craveiro, and P. Verissimo, "Architecting robustness and timeliness in a new generation of aerospace systems," in *Architecting Dependable Systems VII (LNCS 6420)*, A. Casimiro, R. de Lemos, and C. Gacek, Eds. Springer, 2010, pp. 146–170.

[6] J. Yang, H. Kim, S. Park, C. Hong, and I. Shin, "Implementation of compositional scheduling framework on virtualization," in *CRTS '10*, Nov. 2010.

[7] J. Lee, S. Xi, S. Chen, L. T. Phan, C. Gill, I. Lee, C. Lu, and O. Sokolsky, "Realizing compositional scheduling through virtualization," in *RTAS '12*, Beijing, China, Apr. 2011.

[8] B. Kauer, P. Verissimo, and A. Bessani, "Recursive virtual machines for advanced security mechanisms," in *DSN '11 Workshops*, Hong Kong, Jun. 2011.

[9] I. Shin and I. Lee, "Compositional real-time schedulability analysis," in *Handbook of Real-Time and Embedded Systems*, I. Lee, J. Y.-T. Leung, and S. H. Son, Eds. Chapman & Hall/CRC, 2007.

[10] J. Craveiro, J. Rufino, and F. Singhoff, "Architecture, mechanisms and scheduling analysis tool for multicore time- and space-partitioned systems," in *ECRTS '11 WiP*, Jul. 2011.

[11] J. L. Herman, C. J. Kenna, M. S. Mollison, J. H. Anderson, and D. M. Johnson, "RTOS support for multicore mixed-criticality systems," in *RTAS '12*, Beijing, China, Apr. 2011.

[12] J. P. Craveiro and J. Rufino, "Heterogeneous multiprocessor compositional real-time scheduling," in *RTSOPS 2012*, Pisa, Italy, Jul. 2012.

[13] I. Shin, A. Easwaran, and I. Lee, "Hierarchical scheduling framework for virtual clustering of multiprocessors," in *ECRTS '08*, Prague, Czech Republic, Jul. 2008.

[14] A. Easwaran, I. Shin, and I. Lee, "Optimal virtual cluster-based multiprocessor scheduling," *Real-Time Syst.*, vol. 43, pp. 25–59, 2009.

[15] S. Baruah and J. Goossens, "The EDF scheduling of sporadic task systems on uniform multiprocessors," in *RTSS '08*, Barcelona, Spain, Nov./Dec. 2008.

[16] E. Bini, M. Bertogna, and S. Baruah, "Virtual multiprocessor platforms: Specification and use," in *RTSS '09*, Washington, D.C., Dec. 2009.

[17] G. Lipari and E. Bini, "A framework for hierarchical scheduling on multiprocessors: From application requirements to run-time allocation," in *RTSS '10*, Nov./Dec. 2010.

[18] A. Burmyakov, E. Bini, and E. Tovar, "The Generalized Multiprocessor Periodic Resource interface model for hierarchical multiprocessor scheduling," in *RTNS '12*, Nov. 2012.

[19] S. Funk, J. Goossens, and S. Baruah, "On-line scheduling on uniform multiprocessors," in *RTSS '01*, London, UK, Dec. 2001.

[20] S. Baruah, "An improved global EDF schedulability test for uniform multiprocessors," in *RTAS '10*, Stockholm, Sweden, Apr. 2010.

[21] S. K. Baruah, A. K. Mok, and L. E. Rosier, "Preemptively scheduling hard-real-time sporadic tasks on one processor," in *RTSS '90*, Lake Buena Vista, Florida, USA, Dec. 1990.

[22] A. K. Mok, X. A. Feng, and D. Chen, "Resource partition for real-time systems," in *RTAS '01*, Taipei, Taiwan, May/Jun. 2001.

[23] T. P. Baker, "Multiprocessor EDF and Deadline Monotonic schedulability analysis," in *RTSS '03*, Cancun, Mexico, Dec. 2003.

[24] S. Baruah, "Techniques for multiprocessor global schedulability analysis," in *RTSS '07*, Tucson, AZ, Dec. 2007.

[25] M. M. van den Heuvel, R. J. Bril, J. J. Lukkien, and M. Behnam, "Extending a HSF-enabled open-source real-time operating system with resource sharing," in *OSPERT '10*, Brussels, Belgium, Jul. 2010.

[26] J. P. Craveiro, R. O. Silveira, and J. Rufino, "hsSim: an extensible interoperable object-oriented n-level hierarchical scheduling simulator," in *WATERS 2012*, Pisa, Italy, Jul. 2012.

[27] M. Holenderski, R. J. Bril, and J. J. Lukkien, "Grasp: Visualizing the behavior of hierarchical multiprocessor real-time systems," in *WATERS 2011*, Porto, Portugal, Jul. 2011.

APPENDIX

Theorem 4. *Consider the UMPR $\mathcal{U} = (\Pi, \Theta, \pi)$. If π is in fact an identical multiprocessor platform, this transformation becomes equivalent to that in [14] for the MPR.*

Proof: If π is an m-processor identical multiprocessor platform, then all processors have speed 1.0, and $S_\ell(\pi) = \ell$, for all $\ell \leq m$. Hence, $\beta \stackrel{\text{def}}{=} \Theta - S_m(\pi) \cdot \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor = \Theta - m \cdot \left\lfloor \frac{\Theta}{m} \right\rfloor$ and $k \stackrel{\text{def}}{=} \max\{\ell : S_\ell(\pi) \leq \beta\} = \max\{\ell \in \mathbb{N} : \ell \leq \beta\} = \lfloor \beta \rfloor$. These match the definitions in [14]. When we replace $S_m(\pi)$ for m and all s_i's by 1 in the definition of our transformation (Definition 3), we have the following rules:

- if $i \leq k$, $\tau_i^{(\mathcal{U})} = \left(\Pi, \left\lfloor \frac{\Theta}{m} \right\rfloor + 1), \Pi\right)$;
- if $i = k+1$, $\tau_i^{(\mathcal{U})} = \left(\Pi, \left\lfloor \frac{\Theta}{m} \right\rfloor + \beta - S_k(\pi), \Pi\right) = \left(\Pi, \left\lfloor \frac{\Theta}{m} \right\rfloor + \beta - k \cdot \left\lfloor \frac{\beta}{k} \right\rfloor, \Pi\right)$;
- otherwise, $\tau_i^{(\mathcal{U})} = \left(\Pi, \left\lfloor \frac{\Theta}{m} \right\rfloor, \Pi\right)$;

which match the rules of the transformation from MPR to tasks presented in [14]. ■

Lemma 7. *The sum of the execution requirements of all the tasks in $\tau^{(\mathcal{U})}$ is equal to Θ.*

Proof: To prove this lemma, we consider three separate cases:

1) Θ is exactly divisible by $S_m(\pi)$;
2) Θ is not exactly divisible by $S_m(\pi)$, and $S_k(\pi) = \beta$;
3) Θ is not exactly divisible by $S_m(\pi)$, and $S_k(\pi) < \beta$.

Case 1. If Θ is exactly divisible by $S_m(\pi)$, then $\beta = 0$ and $k = 0$. Hence, $\tau^{(\mathcal{U})}$ will have:

- one task ($\tau_1^{(\mathcal{U})} \stackrel{\text{def}}{=} \tau_{k+1}^{(\mathcal{U})}$) with execution requirement $\left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i + \beta - S_k(\pi) = \frac{\Theta}{S_m(\pi)} \cdot s_i$, with $i = 1$, and;
- $m - 1$ tasks, each with execution requirement $\left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i = \frac{\Theta}{S_m(\pi)} \cdot s_i$, with $i \in [2..m]$.

$$\begin{aligned}
\sum_{i=1}^{m} C_i^{(\mathcal{U})} &= \sum_{i=1}^{m} \frac{\Theta}{S_m(\pi)} \cdot s_i \\
&= \frac{\Theta}{S_m(\pi)} \cdot \sum_{i=1}^{m} s_i \\
&= \frac{\Theta}{S_m(\pi)} \cdot S_m(\pi) = \Theta \ .
\end{aligned}$$

Case 2. If Θ is not exactly divisible by $S_m(\pi)$, then $\beta > 0$. Let us consider that k is such that $S_k(\pi) = \beta$ (therefore, $k > 0$). Hence, $\tau^{(\mathcal{U})}$ will have:

- k tasks with execution requirement $\left(\left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor + 1\right) \cdot s_i$, with $i \in [1..k]$;
- one task with execution requirement $\left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i + \beta - S_k(\pi) = \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i$, with $i = k+1$, and;
- $m - (k+1)$ tasks with execution requirement $\left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i$, with $i \in [k+2\,..\,m]$.

$$\begin{aligned}
\sum_{i=1}^{m} C_i^{(\mathcal{U})} &= \left(\sum_{i=1}^{k} \left(\left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor + 1\right) \cdot s_i\right) + \left(\sum_{i=k+1}^{m} \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i\right) \\
&= \left(\sum_{i=1}^{k} \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i\right) + \left(\sum_{i=1}^{k} s_i\right) + \left(\sum_{i=k+1}^{m} \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i\right) \\
&= \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot S_m(\pi) + \underbrace{S_k(\pi)}_{\beta} \\
&= \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot S_m(\pi) + \Theta - \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot S_m(\pi) = \Theta \ .
\end{aligned}$$

Case 3. If Θ is not exactly divisible by $S_m(\pi)$, then $\beta > 0$. Let us consider that k is such that $S_k(\pi) < \beta$ (therefore, $k \geq 0$). Hence, $\tau^{(\mathcal{U})}$ will have:

- k tasks with execution requirement $\left(\left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor + 1\right) \cdot s_i$, with $i \in [1..k]$;
- one task with execution requirement $\left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i + \beta - S_k(\pi)$, with $i = k+1$, and;
- $m - (k+1)$ tasks with execution requirement $\left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i$, with $i \in [k+2\,..\,m]$.

$$
\begin{aligned}
\sum_{i=1}^{m} C_i^{(\mathcal{U})} &= \left(\sum_{i=1}^{k} \left(\left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor + 1\right) \cdot s_i\right) + \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_{k+1} + \beta - S_k(\pi) + \left(\sum_{i=k+2}^{m} \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i\right) \\
&= \left(\sum_{i=1}^{k} \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i\right) + \underbrace{\left(\sum_{i=1}^{k} s_i\right)}_{S_k(\pi)} + \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_{k+1} + \beta - S_k(\pi) + \left(\sum_{i=k+2}^{m} \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot s_i\right) \\
&= \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot S_m(\pi) + \beta \\
&= \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot S_m(\pi) + \Theta - \left\lfloor \frac{\Theta}{S_m(\pi)} \right\rfloor \cdot S_m(\pi) = \Theta \ .
\end{aligned}
$$

Having proven our statement for the three cases that have to be considered, we prove the lemma. ■

Theorem 5. *If taskset $\tau^{(\mathcal{U})}$ is schedulable by* GEDF *on π, then the supply to $\mathcal{C}$ is lower-bounded by* $\mathrm{sbf}_{\mathcal{U}}$.

Proof: For each task $\tau_i^{(\mathcal{U})} = (\Pi, C_i^{(\mathcal{U})}, \Pi)$, not missing a deadline means that task $\tau_i^{(\mathcal{U})}$ receives at least C_i execution units from π within every period of length Π. Overall, we have that tasks in taskset $\tau^{(\mathcal{U})}$ receive at least $\sum_{\tau_i^{(\mathcal{U})} \in \tau^{(\mathcal{U})}} C_i^{(\mathcal{U})} = \Theta$ (Lemma 7) execution units from π within every period of length Π. This is exactly the definition of the supply provided according to a UMPR $\mathcal{U} = (\Pi, \Theta, \pi)$. ■